Ministry of Education and Science of the Russian Federation
Peter the Great St. Petersburg State Polytechnic University
Institute of Computer Sciences and Technologies
Graduate School of Cyber-Physical Systems and Control

Practice Task – Ch 5

Artificial Neural Network
Discipline: Intellectual Computing
22 March 2017

Student Group: 13541/8

Christopher W. Blake

Professor

Kuchmin A.Y

St. Petersburg
2017

Christopher W. Blake

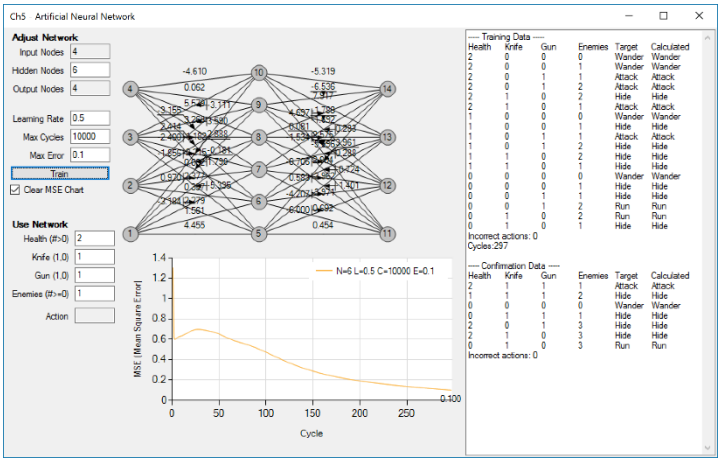
Contents

Christopher W. Blake

Introduction

Chapter 5 of "AI Application Programming" by M. Tim Jones is about artificial neural networks. Artificial neural networks are often used for complex decision making such as decision trees, and generalization between known situations. In this example, a simple game AI is developed. The character of this game must make a decision to attack, wander, hide, or run. This decision is based on the character's health, access to a knife, access to a gun, and the number of enemies.

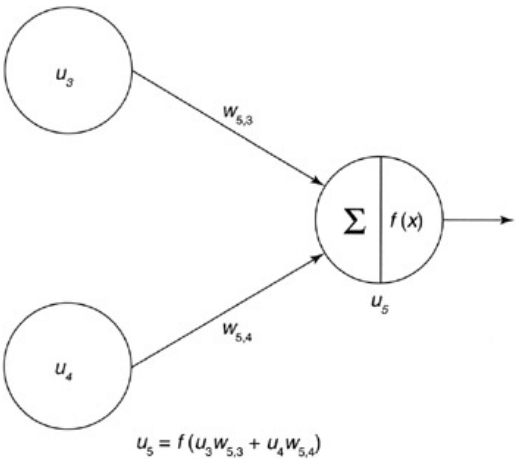
A sample C# program has been created to show this methodology. The program allows the user to specify the network's hidden layers as well as the algorithm parameters (learning rate and learning cycles). The mean square error is plotted vs training cycles and a comparison of the calculated action vs target action is show (with number of mistakes).



Background

Algorithm

A neural network is a very simply system of nodes (shown on the right). Each node is simply a weighted summation of the nodes connected to it and an activation function. Layers of these nodes are connected together to represent complex if-then style logic. This network of nodes is then trained using example input and target data, which uses the below algorithm to adjust the weights. After training, such a network is capable of generalizing between the example inputs.



Backpropagation

1) Select a sample input/target pair of data.
2) Generate values at the output nodes.
   a) Process the input values through the input nodes.
   b) Process the results of the input nodes through the hidden nodes.
   c) Process the hidden nodes to generate the values at the output nodes.
3) Computer error at output layer.
   $\delta_o = (C_i - u_5)u_5(1 - u_5)$ For the output cell
   "C" is the known correct response (target value).
   "u" is the output node.
4) Computer error at the hidden layer
   $\delta_i = (\sum_{m:m>i} w_{m,j}\delta_o)u_i(1 - u_i)$ For all hidden cells
   "w" is the weight of the connection between hidden node "m" and output node "j"
5) Update the weights between hidden layer and output layer.

$$w^*_{i,j} = w_{i,j} + \rho\delta_o u_i$$

"rho" is the learning rate, which is used to encourage convergence.

6) Update the weights between input layer and hidden layer.

$$w^*_{i,j} = w_{i,j} + \rho\delta_i u_i$$

Overfitting Testing

The program allows for chaning of 4 parameters. The number of nodes (N), the learning rate (R), the maximum allowed cycles (C), and the maximum allowed error (E). For comparison the maximum allowed cycles is set very high at 10000. This way the maximum error is always achieve.

By changing the other parameters and comparing training data verus extra confirmation data, situations that cause overfitting can be identified.

Test ID	Nodes (N)	Learning Rate (L)	Max Allowed Error (E)	Errors Training	Errors Confirmation
1	3	0.1	0.4	did not converge	
2	4	0.1	0.4	2	1
3	5	0.1	0.4	2	1
4	6	0.1	0.4	2	1
5	3	0.3	0.4	2	0
6	4	0.3	0.4	2	1
7	5	0.3	0.4	2	1
8	6	0.3	0.4	2	1
9	3	0.5	0.4	2	1
10	4	0.5	0.4	2	1
11	5	0.5	0.4	2	1
12	6	0.5	0.4	2	0
13	3	0.1	0.2	did not converge	
14	4	0.1	0.2	0	0
15	5	0.1	0.2	0	0
16	6	0.1	0.2	0	0
17	3	0.3	0.2	did not converge	
18	4	0.3	0.2	0	0
19	5	0.3	0.2	0	0
20	6	0.3	0.2	0	0
21	3	0.5	0.2	1	1
22	4	0.5	0.2	0	0
23	5	0.5	0.2	0	0
24	6	0.5	0.2	0	0
25	3	0.1	0.1	did not converge	
26	4	0.1	0.1	0	0
27	5	0.1	0.1	0	0
28	6	0.1	0.1	0	0
29	3	0.3	0.1	0	0
30	4	0.3	0.1	0	0
31	5	0.3	0.1	0	0
32	6	0.3	0.1	0	0
33	3	0.5	0.1	did not converge	
34	4	0.5	0.1	0	0
35	5	0.5	0.1	0	0
36	6	0.5	0.1	0	0

Christopher W. Blake

Conclusion

Various configurations of a neural network were created to simulated decision making of a game character. After creation of this neural network, it was tested for overfitting situations. However, no situations were discovered that caused overfitting, with the provided training data and additional confirmation data. It should be noted there were a couple situations that caused errors in the training data but not the confirmation data (tests 5 and 12).

From testing, it can be seen that many configurations work perfectly. The only difference between these configuration is typical speed to convergence, which is anywhere from 50 cycles to 2000 cycles.